

**VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY**  
**UNIVERSITY OF INFORMATION TECHNOLOGY**

**FACULTY OF COMPUTER NETWORKS AND COMMUNICATIONS**



**UIT**  
**TRƯỜNG ĐẠI HỌC**  
**CÔNG NGHỆ THÔNG TIN**

**PROJECT REPORT**

**SUBJECT: WIRELESS EMBEDDED NETWORK SYSTEMS**

**PROJECT: DOOR CONTROL WITH ESP32**

**CLASS: NT131.N12.MMCL**

**MENTOR: PROF. QUAN LE-TRUNG**

***GROUP MEMBER:***

*Phạm Anh Tú – 20522100*

*Nguyễn Đình Thanh Ngân – 20521646*

## Table of Contents

<b>I. Overview.....</b>	<b>3</b>
1. The problem .....	3
2. Solution .....	3
<b>II. System design and implementation .....</b>	<b>4</b>
1. Activity Flowchart .....	4
2. Devices .....	5
3. Diagrams.....	6
4. Code overview.....	7
5. Cloud Server .....	9
<b>III. Result &amp; Demo .....</b>	<b>11</b>
1. Features .....	11
2. System Log .....	12
3. Demo .....	13
<b>IV. Issues .....</b>	<b>13</b>
<b>V. References .....</b>	<b>14</b>

# I. Overview

## 1. The problem

Today, smart homes are becoming increasingly popular due to the convenience they provide. It consists of many smart devices linked together to bring convenience to users, making life easier. In particular, the smart door is a typical and important device in the smart home system. Why? For example, after a tiring day at work, you come home and stand at the door. It's terrible if you have to rummage through your bag just to find a key for the sole purpose of opening the door and entering the house. It will be worse when the key is still on the desk at the company or is sometimes lost. Another example is that your house has many rooms and doors; the total number is up to 20. Every time you need to open a certain door, you will take out the set of keys, find out which is the correct key, and open the door to that room. The above simple reasons have made smart doors important and necessary in the smart home system.

## 2. Solution

For the reasons listed above, we propose to design and deploy a simple door control system via Internet-connected phone that replaces the physical key. The system can be deployed in many different cases, such as offices, gardens, garages, or private houses. Door data is stored in the cloud, making it easy for users to manage and control it from anywhere and reducing security risks. The system uses ESP32 to control the opening and closing of the electric door, along with an RFID reader in case the WiFi connection fails. The advantages are low cost and ease of deployment, use, and upgrade.

## II. System design and implementation

### 1. Activity Flowchart

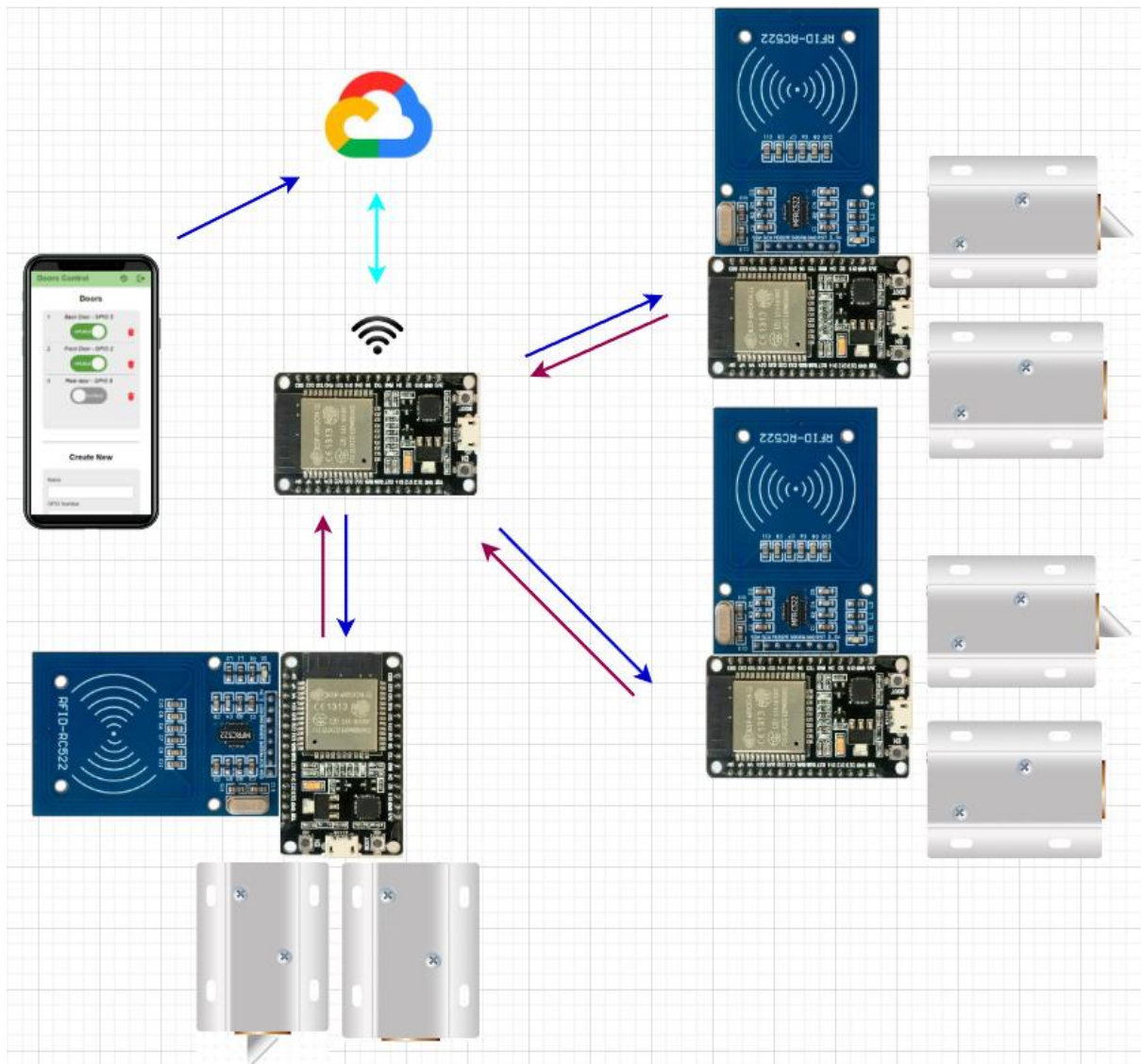

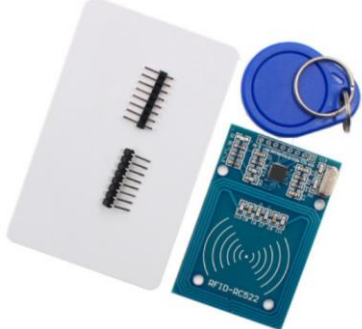

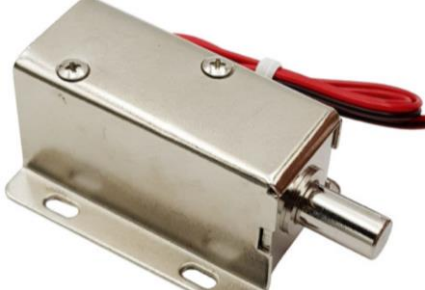


Figure 1. Activity flowchart

- Users access the web and make a change to the door state, door data will be refreshed every 3 seconds.
- The master will make an HTTP request to get data from the server every 3 seconds. It then analyzes and sends open/close commands to its slaves. The slaves receiving the command will only open or close the door on request when the request belongs to the door they are managing. Master and slave communication is based on ESP-Now technology.
- When using the RFID Reader, the slave will pause to receive the command from the master and execute the command from the reader (the door will close automatically after 8 seconds). Also send a message to the master about the status of the door.
- The master receiving messages from the slave will also update the door status on the server.

## 2. Devices

<i>Name</i>	<i>Description</i>	<i>Quick look</i>
<b>ESP32</b>	ESP32 is a low-cost, low-power microcontroller with integrated Wi-Fi and Bluetooth. It's a successor to the ESP8266. Used as main controller.	
<b>RFID Reader Kit</b>	RFID Reader: Using a unique identification number for each tag or card, an RFID reader device scans the RFID tag or card to retrieve the identifying information. Used as an alternative way to open doors.	
<b>Relay 12V</b>	Use low voltage to toggle on or off the high voltage. Used as a door activator.	
<b>Door Lock</b>	Toggled on or off by relay.	

- The system is divided into two main modules: the master module consists of a single ESP32 as controller, and the door module includes an ESP32 connected with an RFID reader and door lock.
- ESP32 is used as the main controller because of its low cost, high performance, low power consumption, and especially WiFi connection support.
- The RFID reader is used as a tentative way of opening the door in case WiFi encounters a problem or can't use the phone to open the door.



### 3. Diagrams

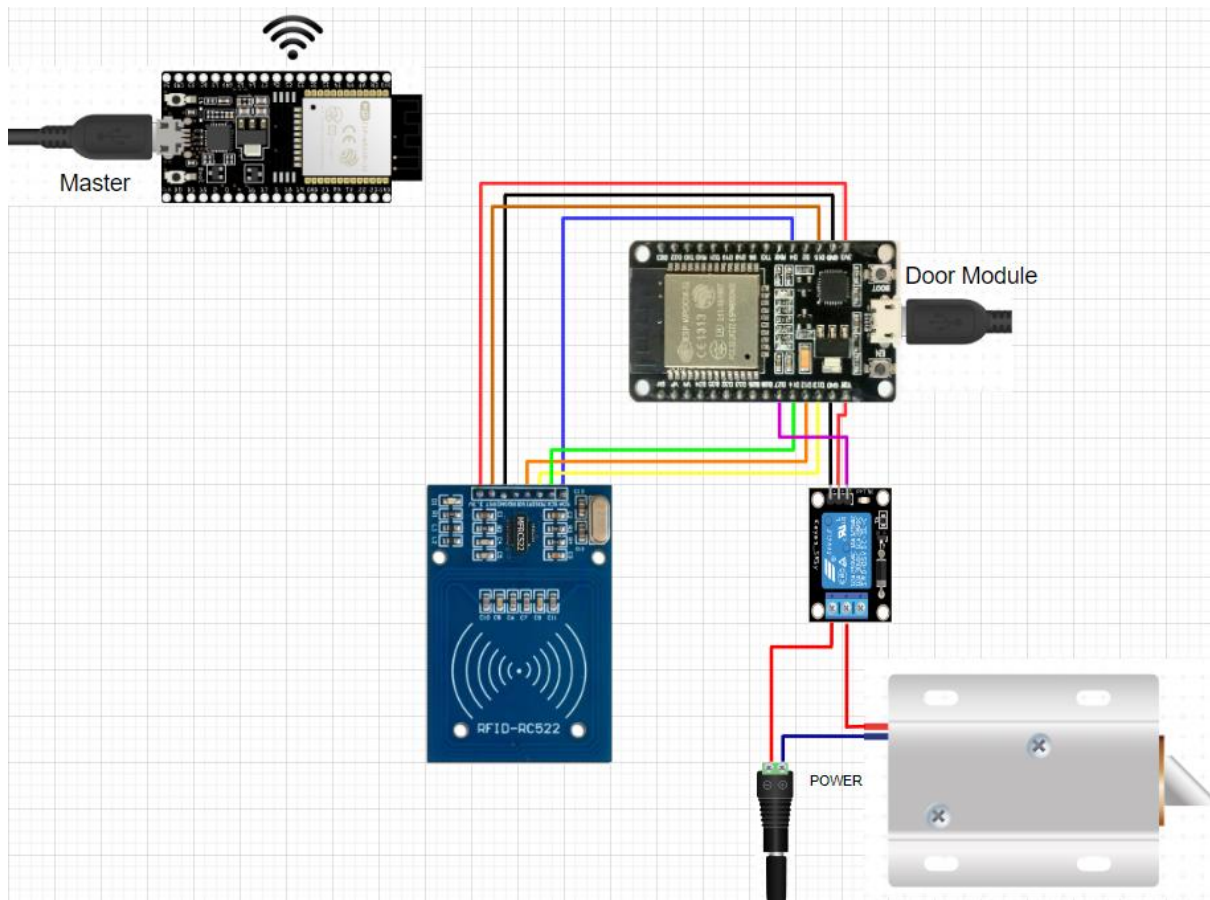


Figure 2. Circuit diagram

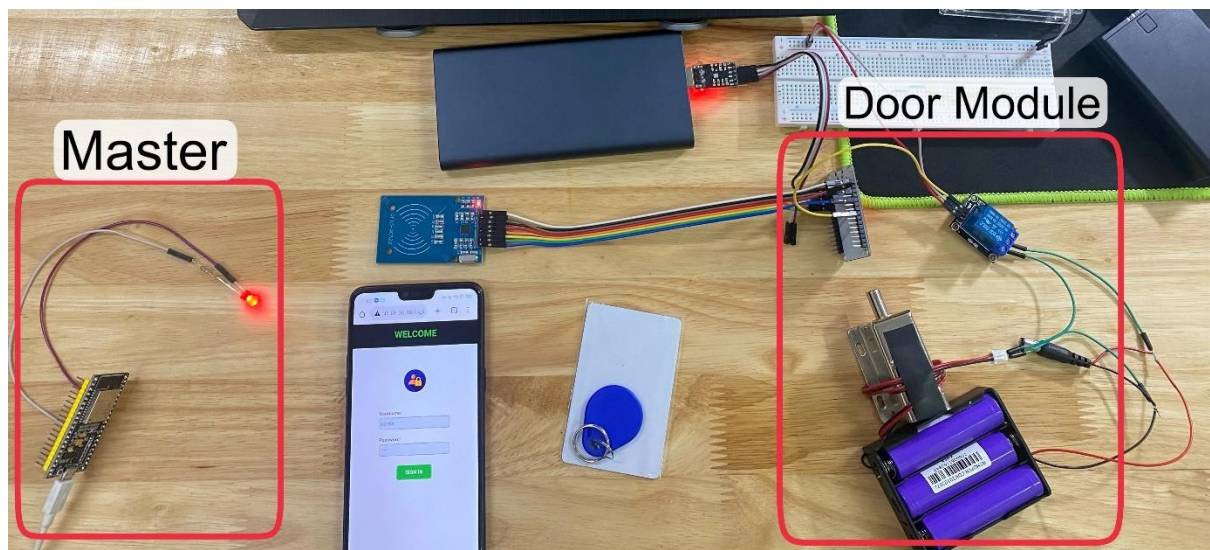


Figure 3. Actual diagram

## 4. Code overview

### a. Master controller

#### 1. Setup Function

```
if (!EEPROM.begin(EEPROM_SIZE)) {  
    Serial.println("Failed to init EEPROM");  
    delay(1000);  
}  
else  
{  
    ssid = readStringFromFlash(0);  
    password = readStringFromFlash(40);  
    Serial.println(ssid);  
    Serial.println(password);  
}
```

- Initialize EEPROM and read the last saved WiFi information.
- EEPROM save data permanently, data can be change programmatically and data will not lose when power off.

```
WiFi.begin(ssid.c_str(), password.c_str());  
delay(3000);  
if (WiFi.status() != WL_CONNECTED)  
{  
    WiFi.beginSmartConfig();  
    Serial.println("Waiting for SmartConfig.");  
    while (!WiFi.smartConfigDone()) {  
        Serial.print(".");  
        BlinkLED();  
        delay(500);  
    }  
}
```

- Initialize WiFi connection, ESP-Touch smart config will start to reconfig the WiFi if the WiFi connection failure.

```
InitESPNow();  
esp_now_register_send_cb(OnDataSent);  
esp_now_register_recv_cb(OnDataRecv);
```

- Initialize ESP-NOW and register 2 callback functions.
- ESP-NOW is used for communication between controller and receiver. 2 callback functions will triggered when data sent or received.
- After all initialization is complete, master scans nearby slaves for connection.

#### 2. Loop Function

```

String outputsState = getRequest();
JSONVar myObject = JSON.parse(outputsState);
if (JSON.typeof(myObject) == "undefined") {
    Serial.println("Parsing input failed!");
    return;
}

JSONVar keys = myObject.keys();
for (int i = 0; i < keys.length(); i++) {
    JSONVar value = myObject[keys[i]];
    Serial.print("GPIO: "); Serial.print(keys[i]);
    Serial.print(" - SET: "); Serial.println(value);
    outMsg.doorNum = atoi(keys[i]);
    outMsg.doorStatus = atoi(value);
    sendMsg();
}

```

- The loop function will get door's data from server and send it to slaves every 3 seconds.

## b. Slave

### 1. Setup Function

```

SPI.begin(SCK_PIN, MISO_PIN, MOSI_PIN, SS_PIN);
mfrc522.PCD_Init();
Serial.println("RFID OK!");

```

- Initialize RFID Reader.

```

InitESPNow();
esp_now_register_send_cb(OnDataSent);
esp_now_register_rcv_cb(OnDataRecv);

```

- Initialize ESP-NOW and register 2 callback functions. Callback function will use the information from master to toggle the door.

### 2. Loop Function



```

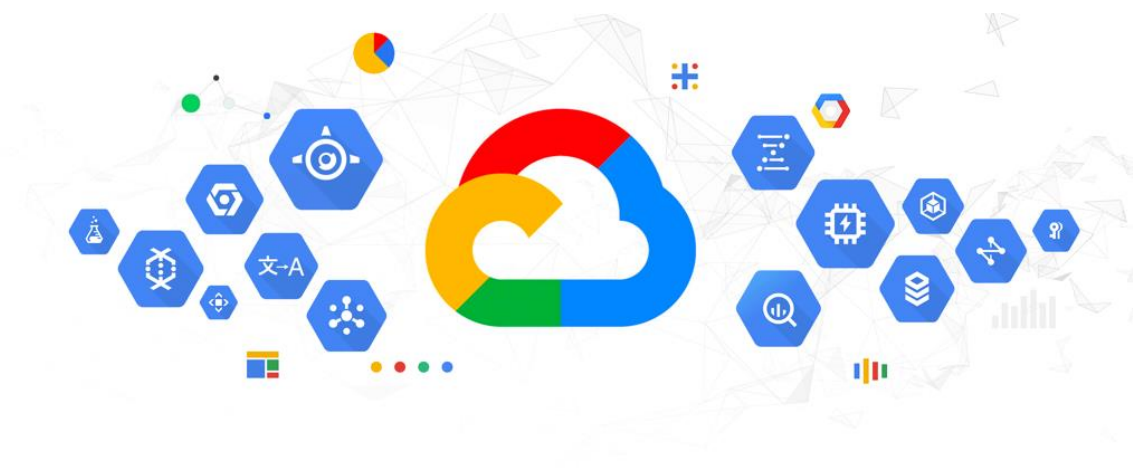
if (mfrc522.PICC_IsNewCardPresent() && mfrc522.PICC_ReadCardSerial())
{
    String content= "";
    for (byte i = 0; i < mfrc522.uid.size; i++)
    {
        content.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " "));
        content.concat(String(mfrc522.uid.uidByte[i], HEX));
    }
    content.toUpperCase();
    if (content.substring(1) == "BD 94 FC 30") Authorized();
    else Unauthorized();
    lastRead = currMillis;
}

```

- The loop function will read the information from the RFID reader to decide open the door or not.

## 5. Cloud Server




- Use Google Cloud Platform to deploy website, run simple LEMP server with Nginx, MySql and PHP.



L E M P  
 linux nginx mysql php

- **Must-have features:** Store door's data, create/ delete door, toggle door open/close.

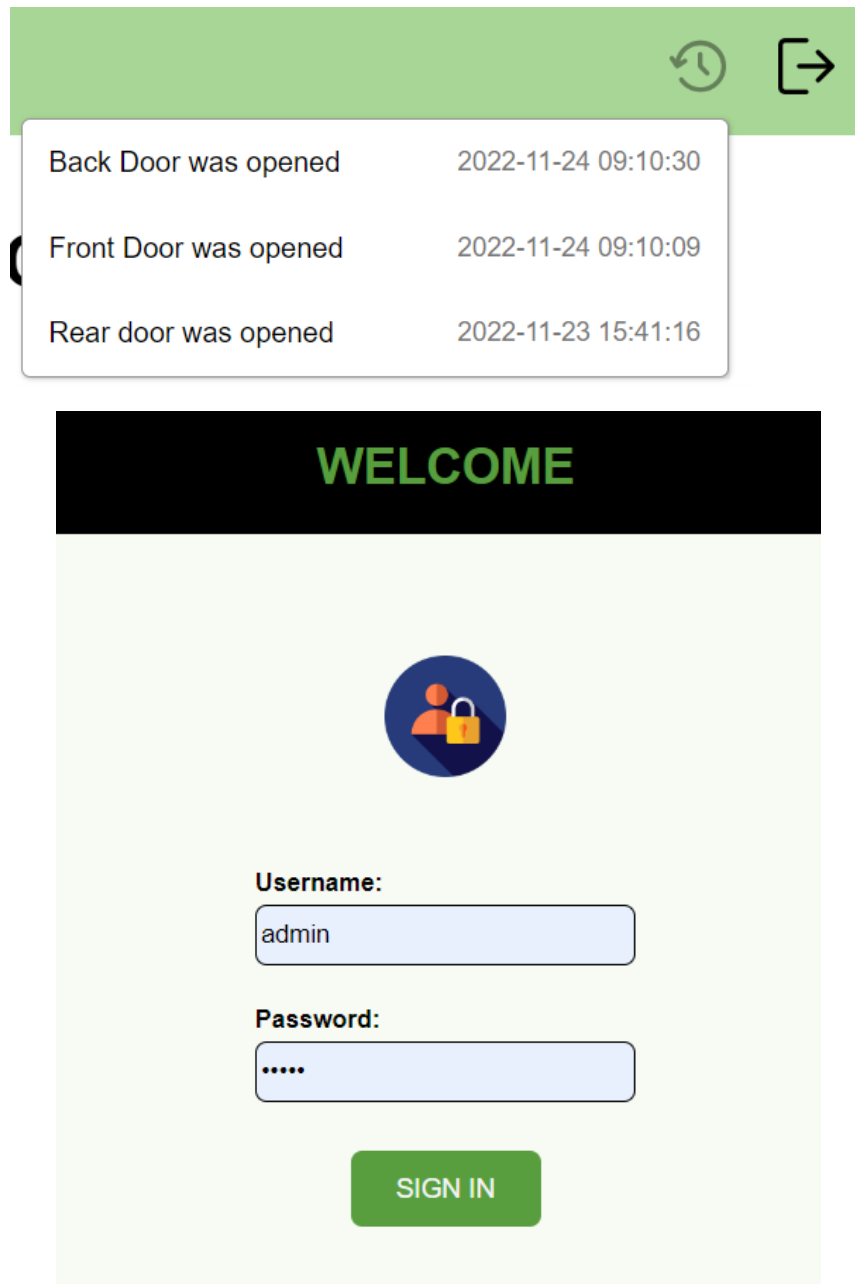
## Doors

1	Back Door - GPIO 3	<input checked="" type="checkbox"/> OPENED	
2	Front Door - GPIO 2	<input checked="" type="checkbox"/> OPENED	
3	Rear door - GPIO 5	<input type="checkbox"/> CLOSED	

## Create New

Name:	<input type="text"/>
GPIO Number:	<input type="text"/>
Current Status:	<input type="text" value="0 = Close"/>
<input type="button" value="CREATE"/>	

- **Optional features:** user authentication (login/logout), view open history, cross-platform use.

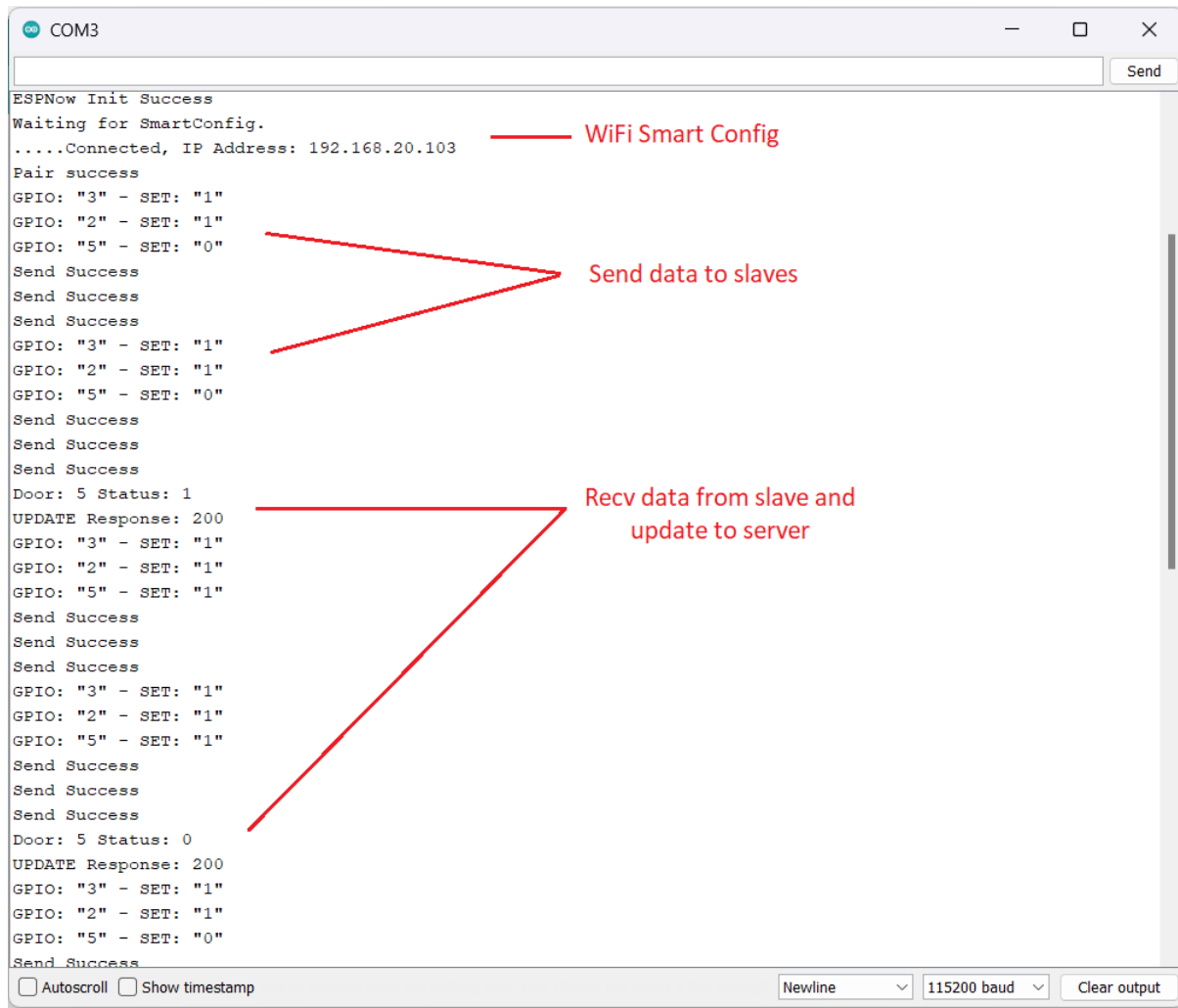


### III. Result & Demo

#### 1. Features

- Control door from everywhere (Web access)
- Add/remove door easily
- Open directly with RFID Card
- Configure ESP32 WiFi with Smartphone (Hold boot button more than 3 seconds to erase old config)

## 2. System Log



COM3

```
ESPNow Init Success
Waiting for SmartConfig.
.....Connected, IP Address: 192.168.20.103
Pair success
GPIO: "3" - SET: "1"
GPIO: "2" - SET: "1"
GPIO: "5" - SET: "0"
Send Success
Send Success
Send Success
GPIO: "3" - SET: "1"
GPIO: "2" - SET: "1"
GPIO: "5" - SET: "0"
Send Success
Send Success
Send Success
Door: 5 Status: 1
UPDATE Response: 200
GPIO: "3" - SET: "1"
GPIO: "2" - SET: "1"
GPIO: "5" - SET: "1"
Send Success
Send Success
Send Success
GPIO: "3" - SET: "1"
GPIO: "2" - SET: "1"
GPIO: "5" - SET: "1"
Send Success
Send Success
Send Success
Door: 5 Status: 0
UPDATE Response: 200
GPIO: "3" - SET: "1"
GPIO: "2" - SET: "1"
GPIO: "5" - SET: "0"
Send Success
```

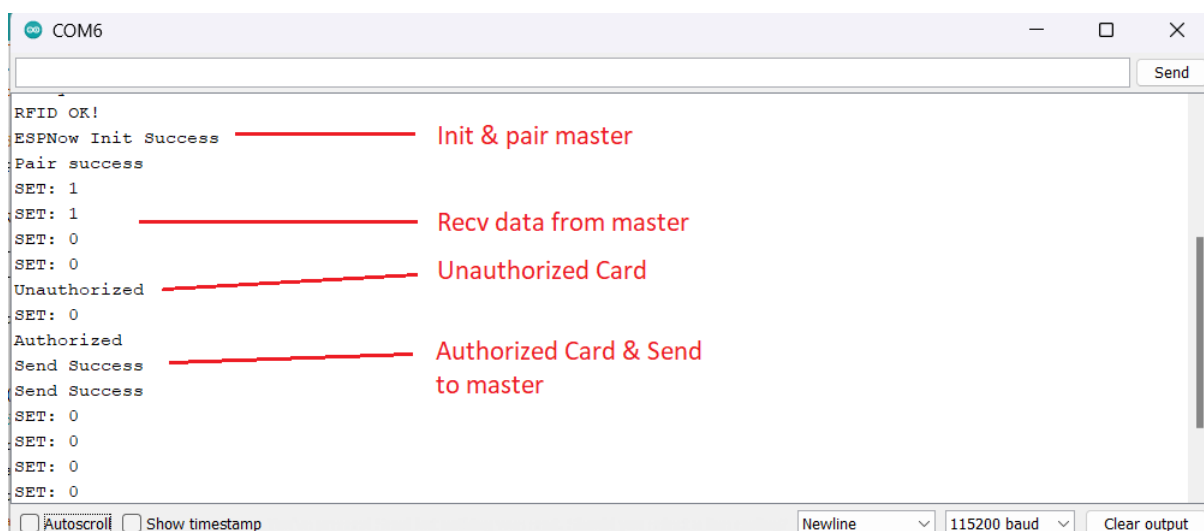
WiFi Smart Config

Send data to slaves

Recv data from slave and update to server

☐ Autoscroll ☐ Show timestamp Newline 115200 baud Clear output

*Master's log*



COM6

```
RFID OK!
ESPNow Init Success
Pair success
SET: 1
SET: 1
SET: 0
SET: 0
Unauthorized
SET: 0
Authorized
Send Success
Send Success
SET: 0
SET: 0
SET: 0
SET: 0
```

Init & pair master

Recv data from master

Unauthorized Card

Authorized Card & Send to master

☐ Autoscroll ☐ Show timestamp Newline 115200 baud Clear output

## *Slave's log*

### 3. Demo

- Some demonstration videos are put in the Google Drive link below, each demonstration video for one or several features.
- [https://drive.google.com/drive/folders/1eXkixu5J1AuHHsIRpy9LEWdOEmsdD0u2?usp=share\\_link](https://drive.google.com/drive/folders/1eXkixu5J1AuHHsIRpy9LEWdOEmsdD0u2?usp=share_link)

## IV. Issues

### 1. RFID Reader sometimes not working properly

- Cause: The door lock uses power directly from the ESP32 which makes the unstable voltage.
- Solve: Use two separate power lines for ESP32 and door lock.

### 2. Difficult to change wifi information

- Cause: WiFi information was pre-programmed in the code when programming.
- Solve: Add ESP-Touch smartconfig function and user configurable from Smartphone.

### 3. Can not work when there's a power outage

- Solve: Use rechargeable battery to power door module and door lock, use RFID as alternative access way.

### 4. Wifi channel conflict

- Cause: Home WiFi, Master module, Door module is not in the same channel.
- Solve (temporary): configure home wifi channel to work on the same channel as the module's channel.
- Future: configure the active channel of the modules dynamically according to the active channel of home wifi.

### 5. Delay between actions

- Cause: There is a 3-second break between requests, with a maximum delay of 3 seconds, for the purpose of reducing overheating on the controller, reducing network pressure, and keeping the system stable.

## V. References

[Welcome to ESP32 Arduino Core's documentation — Arduino-ESP32 2.0.2 documentation \(readthedocs-hosted.com\)](#)

[ESP32 IDF — Tài liệu ESP32 1.0](#)

[Lập trình ESP32 Smartconfig thiết lập wifi qua app \(khuenguyencreator.com\)](#)

[ESP32 - RFID/NFC | ESP32 Tutorial \(esp32io.com\)](#)

[Control ESP32 and ESP8266 GPIOs from Anywhere in the World | Random Nerd Tutorials](#)